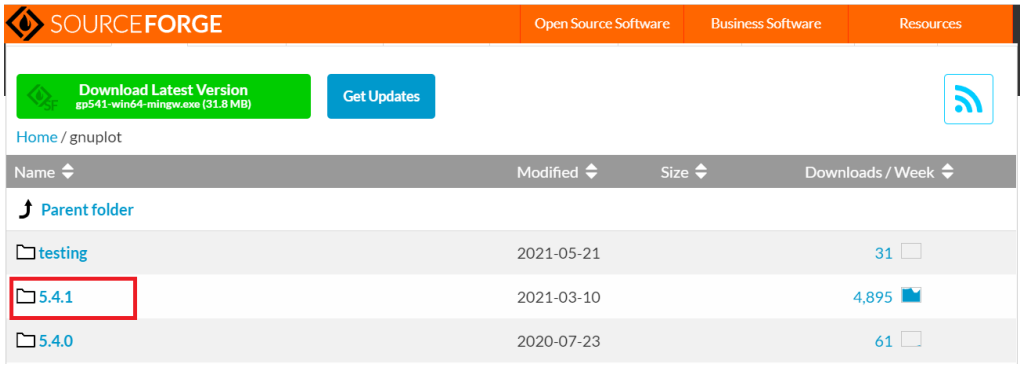
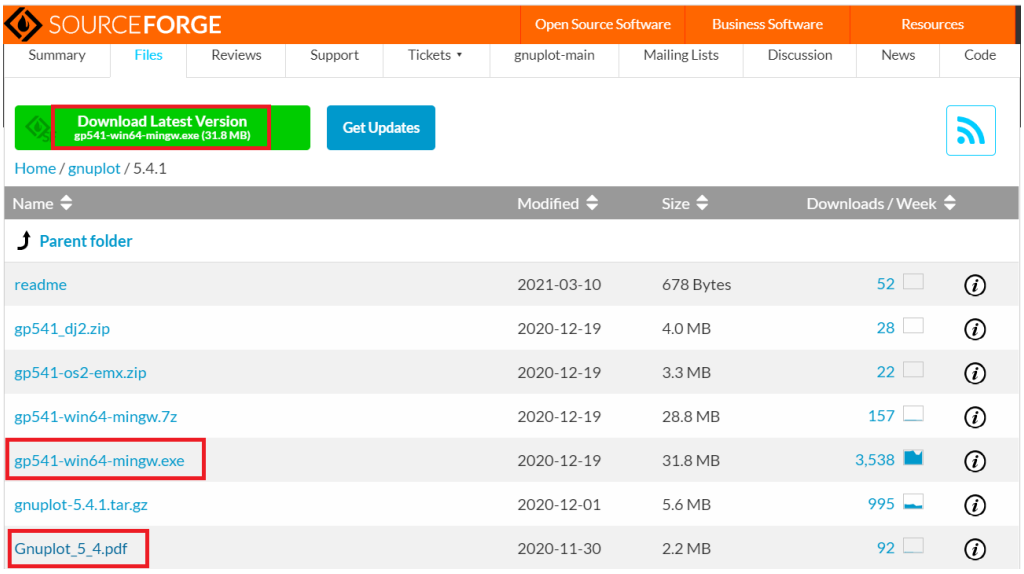
NOTE: This is a manual to install and use Gnuplot for Windows OS 64-bit. For other systems, the processes are similar. Please note that the aim of this manual, eventually, is to allow students to do basic plotting in Gnuplot with C++. If you find any mistake or if you have any comment, please let me know by contacting me at mohdalmie@ukm.edu.my.

A) Installing gnuplot

1. Install gnuplot from http://www.gnuplot.info/download.html. Click on Primary download site on SourceForge. You will see something similar as shown below. The current **stable** version as of 2nd May 2021 is version 5.4.1.

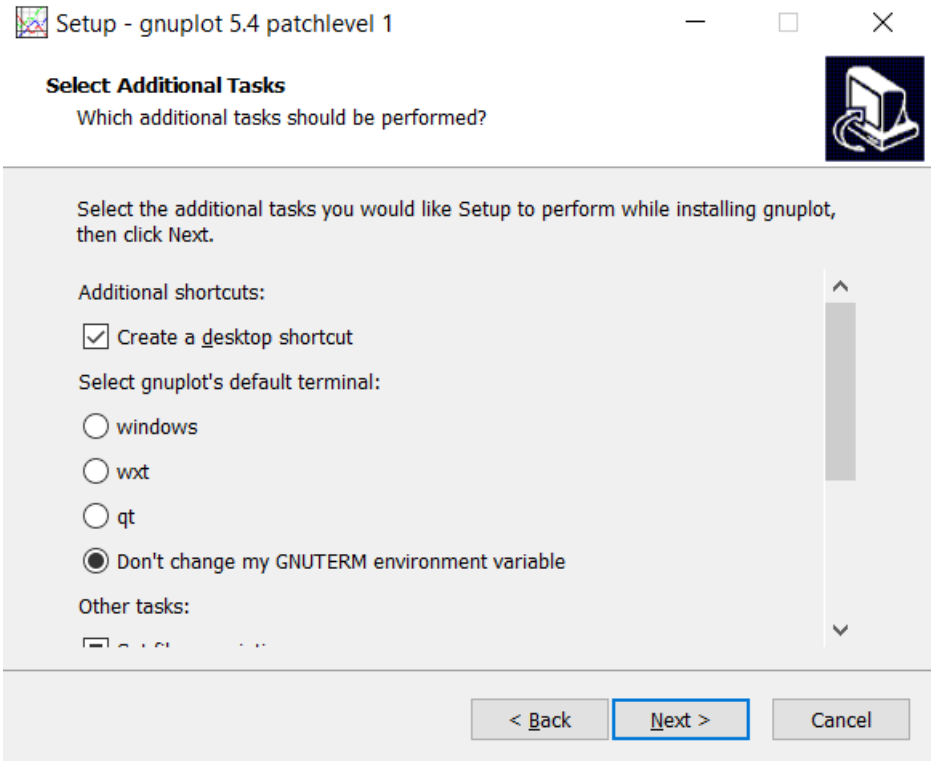


2. Click on the folder 5.4.1. You will then see the files as shown below. Note that you can directly download the installer *gp541-win640mingw.exe*. The file *Gnuplot_5_4.pdf* contains information on gnuplot and how to use it. Please have a look when you have time.

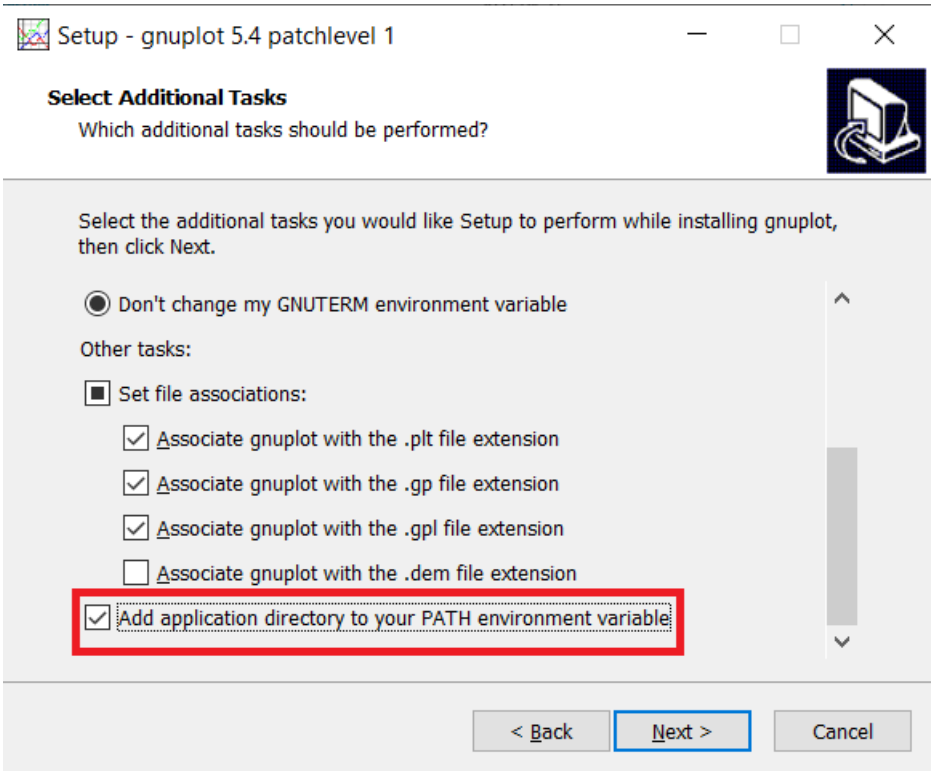


3. Download the installer *gp541-win640mingw.exe* and start installing gnuplot.

4. During the installation, you will see a window displaying something similar like this:

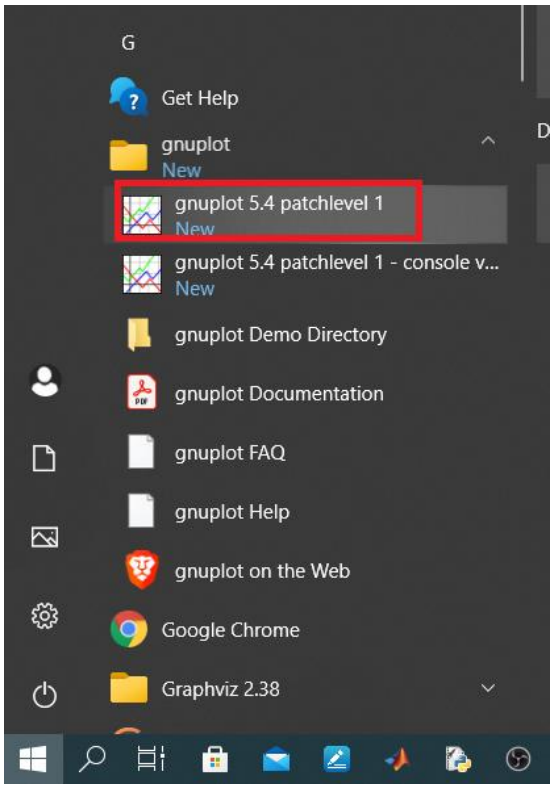


and when scrolled gives

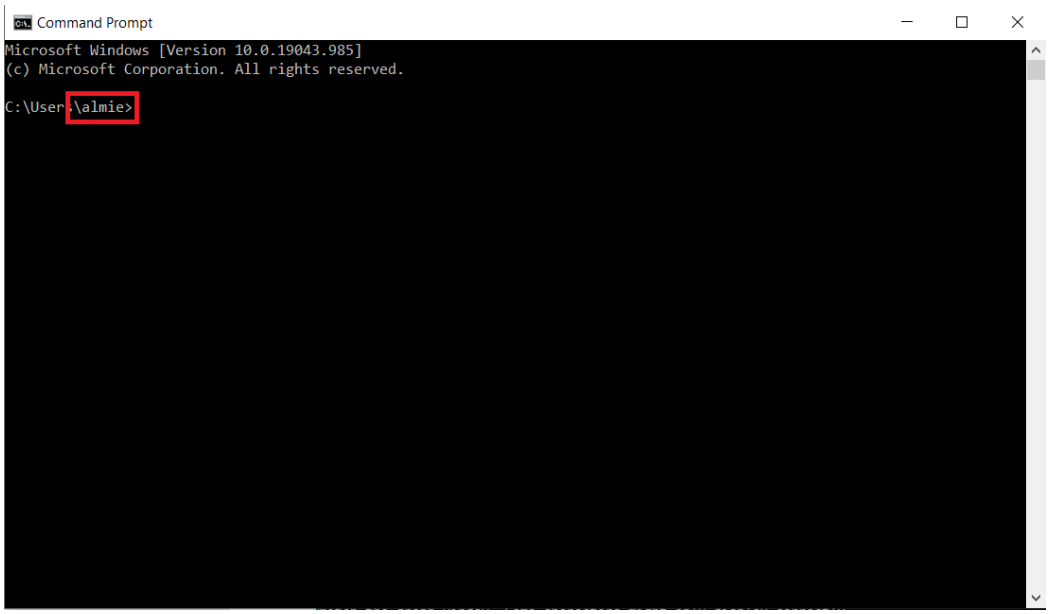


Do not forget to choose "Add application directory to your PATH environment variable".

5. Continue installation.
6. Once finished, you can find gnuplot from your start button. Mine is shown below.

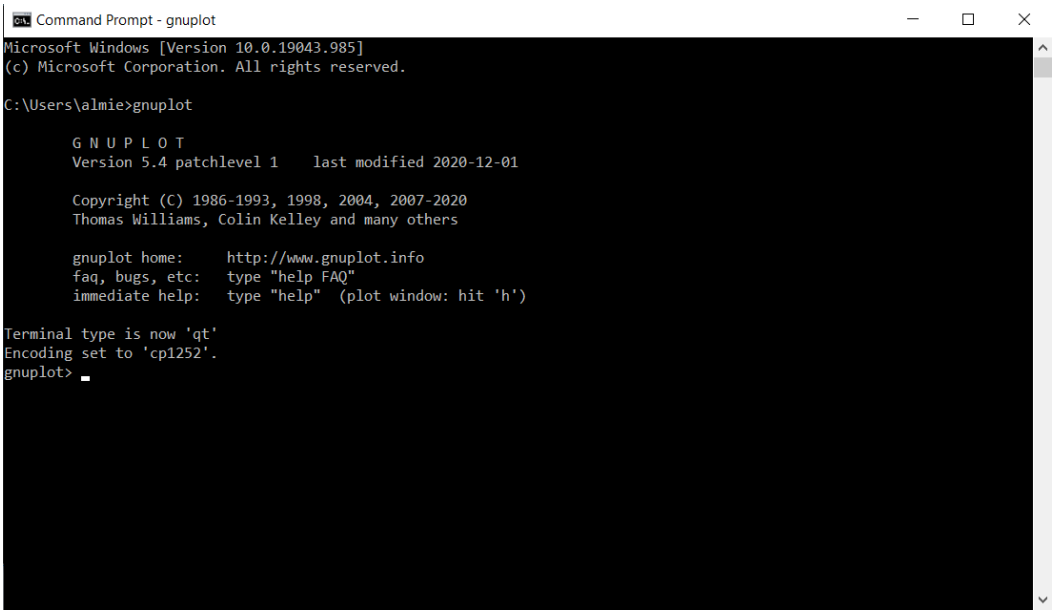


7. You can also open your "cmd" by hitting *Windows + S* followed by typing *cmd* (or try search for *cmd*). This will give a display as follows:



The name shown in the prompt depends on your computer's username. Mine is *almie*.

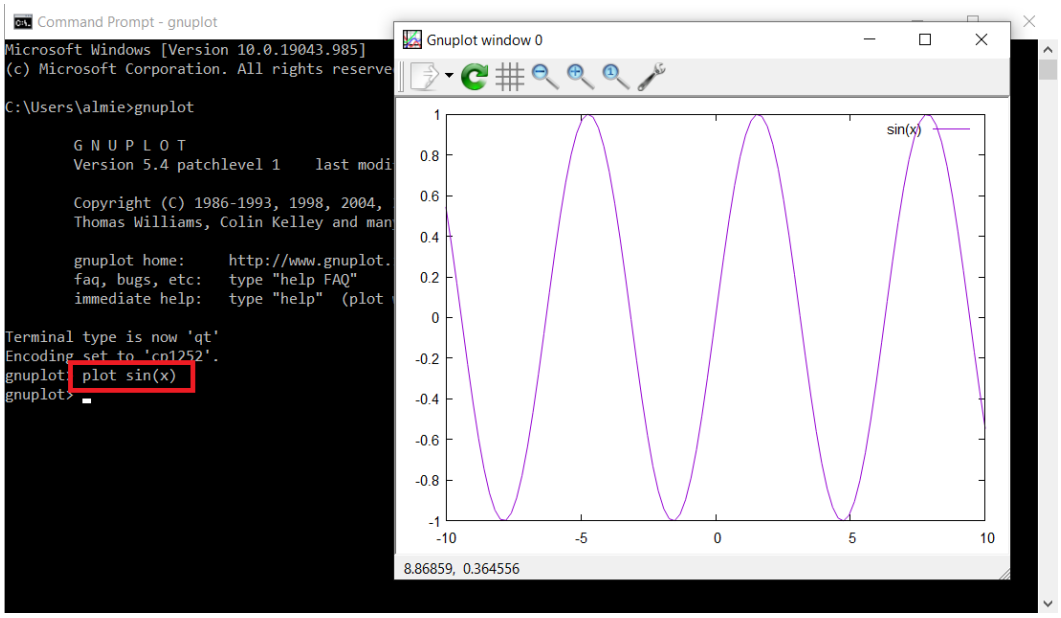
8. Then type *gnuplot*. Something like this will appear:



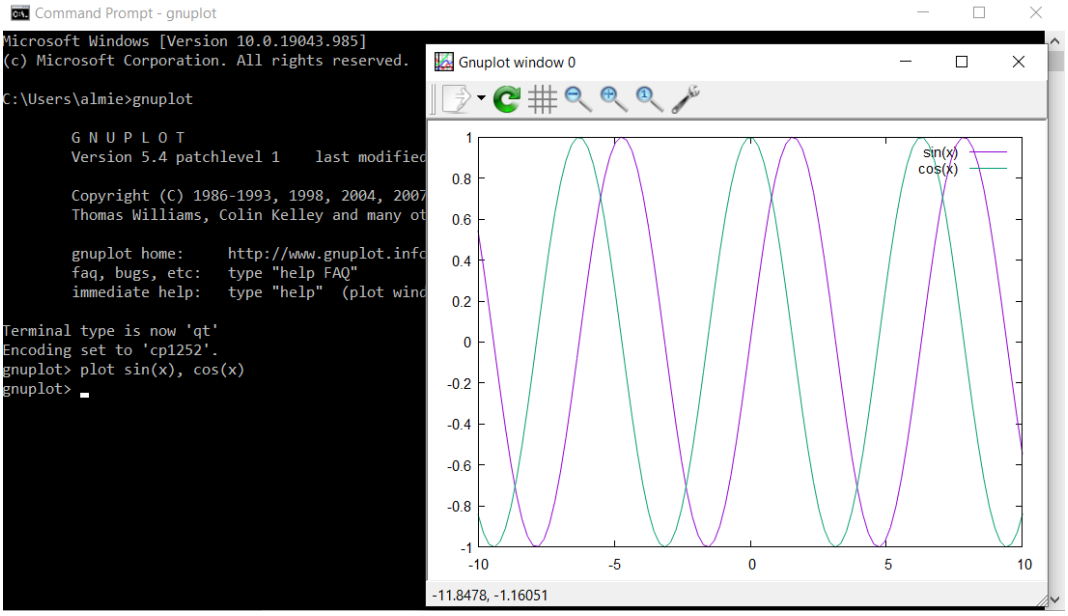
9. This means that gnuplot has been successfully installed in your system.

B) Plotting functions with gnuplot

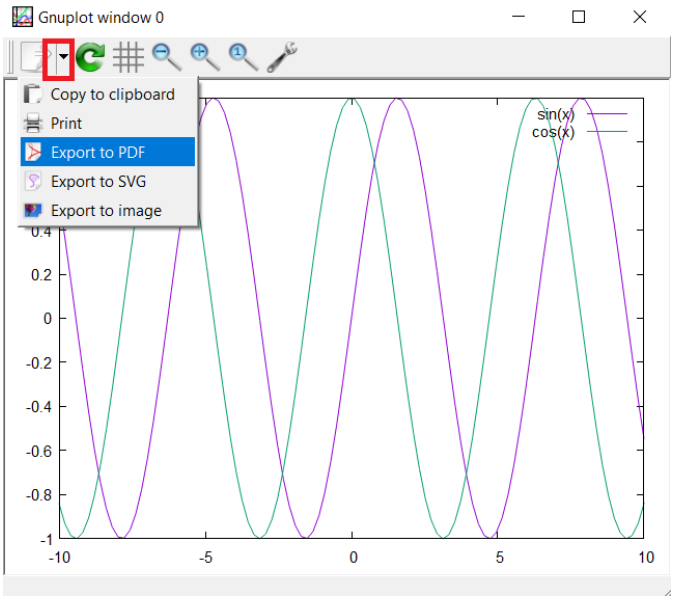
1. Now you can do some plotting, for example by typing *plot sin(x)* and hit enter. This will give you a graph of sin(x).



2. You can plot more than one curves simultaneously by typing *plot sin(x), cos(x)* and hit enter (don't forget to separate both functions by a comma). This will give you a graph of sin(x) and cos(x).



3. Note that you can save the plot by clicking the dropdown arrow.



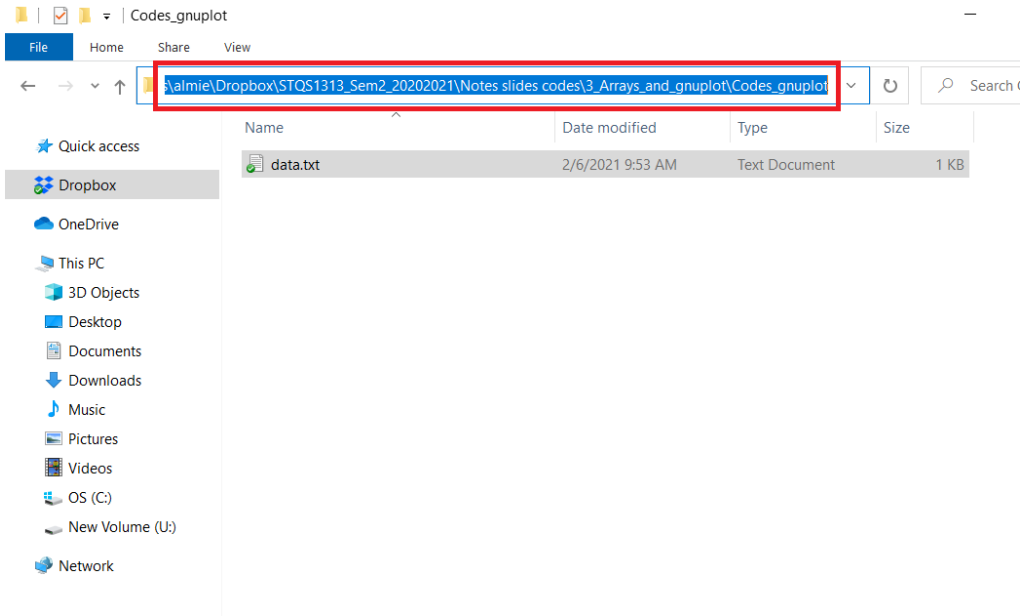
4. Besides the manual *Gnuplot_5_4.pdf*, you can have a look at many tutorials available out there, for example in here https://people.duke.edu/~hpgavin/gnuplot.html or https://www.cs.hmc.edu/~vrable/gnuplot/using-gnuplot.html and http://www.gnuplotting.org/tag/configuration/.
5. To exit gnuplot, type *quit* or simply *q*. Then the gnuplot's prompt will disappear.
6. Then, click x to exit the *cmd*. Alternatively, you can type *exit*.

C) Plotting some data with gnuplot

1. Let's create some data (two-column data) in a file named *data.txt* which contains data as shown below. This example is taken from here https://people.duke.edu/~hpgavin/gnuplot.html.

```
# This file is called data.txt
# Force-Deflection data for a beam and a bar
# Deflection    Col-Force       Beam-Force
0.000              0              0
0.001            104             51
0.002            202            101
0.003            298            148
0.0031           290            149
0.004            289            201
0.0041           291            209
0.005            310            250
0.010            311            260
0.020            280            240
```

2. Save the file.
3. Note that the symbol # at the first three lines of the data will be ignored by gnuplot when it reads the data.
4. Check the location (or path) of your data file. This can be done by clicking the address bar as shown below.

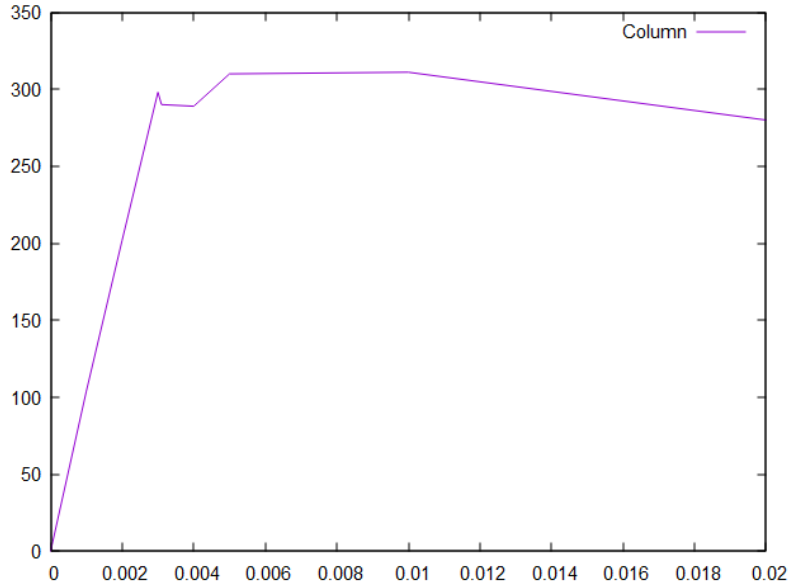


5. We can copy the location (or path) of the file. Let say we name this location as *path*.
6. Then, open gnuplot (see Steps 6-8 from *section A: Installing gnuplot* if you are unsure).
7. From gnuplot, type

   *plot "**path**/data.txt" using 1:2 title 'Column' with lines*

8. Note that the *path* is needed in order to tell gnuplot the location of the data file. You need to make sure the *path* always contains *slash*es /, rather than *backslash*es \.
9. Note that *using 1:2* means we are using column 1 of the data as *x-data* and column 2 as y-data. The *title 'Column'* lets us name the curve in the indicator/legend as *Column*, while *with lines* allows us draw the curve in the form of a line.

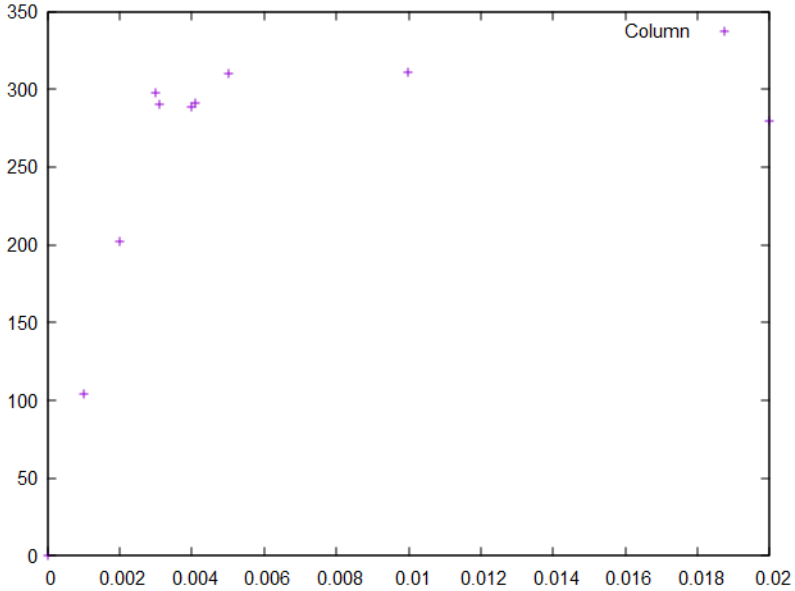
10. **Another important thing is we can replace the double quotes with single quotes**. This is useful particularly in Section J, i.e. when we use gnuplot through C++ to do plotting.
11. Basically, the plot should be similar to what we have below.



12. We can try changing the code (remove *title 'Column'* or *with lines*) to see any difference. For example, typing

    plot "path/data.txt" using 1:2 title 'Column'

    gives the below plot. There are only points and no line.



13. We can also use some **simplifications**. For example, the command

    plot "path/data.txt" using 1:2 title 'Column' with lines

    can be replaced by

    plot "path/data.txt" u 1:2 t 'Column' w l

D) Using loadpath

1. Note that in previous section, we append the path to the file name.
2. Alternatively, we can use loadpath to avoid the need to append path for many times. This is done as follows:

    *set loadpath "**path**"*
    plot "data.txt" u 1:2 t 'Column' w l

3. We can check the current loadpath by typing *show loadpath*

E) Making gnuplot prompt (working directory) to be at the same location as data file's.

1. Another easy way to work with gnuplot and data file is to make sure that the location of gnuplot's prompt to be the same as the location of the data file.
2. This requires us to copy the location of the data file, as done in Step 4 from *section C: Plotting some data with gnuplot.*
3. Now, let say we name the location as *path*.
4. Open gnuplot (see Steps 6-8 from *section A: Installing gnuplot* if you are unsure).
5. Change the location of gnuplot's prompt to the location of the data file. This can be done by typing in gnuplot

    cd "*path*"

    You need to make sure that the path always contains *slash*es /, not *backslash*es \.
6. Now, check the location of your gnuplot's prompt by typing *pwd*. It will output the same name as *path* that you have entered before.
7. Now, from gnuplot, type

    plot "data.txt" u 1:2 t 'Column' w l

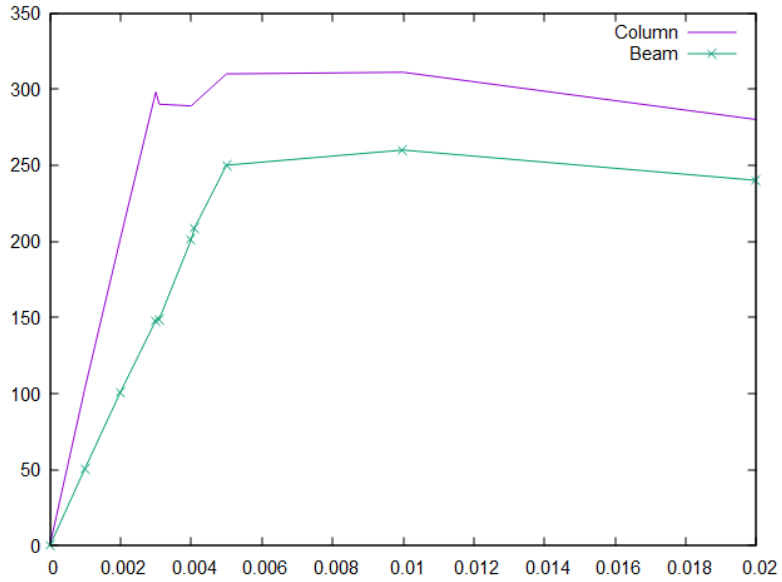
    to produce the same plot as above. Note that you don't need to append the *path* before *data.txt*, which is slightly different from that shown in Step 12 of the previous section.

F) Using other plot properties

1. Since *data.txt* has 3 columns, we can utilise the 3rd column. We assume that the 1st column represents independent variables (x-data) while both 2nd and 3rd columns represent dependent variables (y-data).
2. So, just type in gnuplot (where *lp* represent *linespoints*)

    plot "data.txt" u 1:2 w l t 'Column', "data.txt" u 1:3 w lp t 'Beam'

to produce the plot as shown below.



3. Again, we can remove the *path* from the above command if gnuplot's prompt is in the same location as the data file.
4. This is a very beautiful plot. Again, you can refer to the manual *Gnuplot_5_4.pdf*, or many other tutorials available out there.
5. For example, we can set various plot property at any time by using
   - *set xrange [0:10]*
   - *set xrange [0:400]*
   - *set xlabel "x value"*
   - *set ylabel "y value"*

6. After changing/adding some element in the plot, we need to update the current plot by typing *replot*.

G) Plotting with gnuplot using script

1. Oftentimes, we need to re-use certain plot properties repeatedly. So, it's nice to be able to save the 'format' and call them quickly whenever we need them.
2. This can be done by putting every line of codes in a file, say *plotcode.dat* or *plotcode.txt* or even *plotcode.gnuplot*, and call it from gnuplot by typing *load 'plotcode.dat'* or with double quotes *load "plotcode.dat"*.
3. The script can be created, for example, by using notepad.
4. Assume that we have the same data file as what have been used before, *data.txt*. Then we can create another file for the script in the **same location** where the data file is in, named as *myscript.txt*. The script contains gnuplot codes as follows:

```
plot "data.txt" u 1:2 w l t 'Column', "data.txt" u 1:3 w lp t 'Beam'
set xrange [0:0.02]
set yrange [0:400]
set xlabel "Deflection"
set ylabel " Col-Force/Beam-Force"
replot
```

5. Then we can run the code *myscript.txt* to plot the data saved in *data.txt* by typing in the gnuplot's terminal:

   load "myscript.txt"

6. We can change the order of the lines of the script, starting with setting other properties followed by plotting. This is shown below.

   set xrange [0:02]
   set yrange [0:100]
   set xlabel "Deflection"
   set ylabel " Col-Force/Beam-Force"
   plot "data.txt" u 1:2 w l t 'Column', "data.txt" u 1:3 w lp t 'Beam'

   which avoids the use of *replot* at the end of the script.

7. Of course the codes above assume that the gnuplot's prompt is at the same location as the script and data file.
8. As mentioned in *Section D: Using loadpath*, we can use *loadpath* when the gnuplot's prompt is in different location than the script and data file. In this case, we need to type:

   *set loadpath "**path**"*
   load "myscript.txt"

H) Plotting without entering gnuplot

1. Just in case if we want to use gnuplot from the cmd (without typing gnuplot at the beginning), we need to type

   gnuplot -p "myscript.txt"

   assuming the cmd's prompt is in the same location as that of the script and data files.

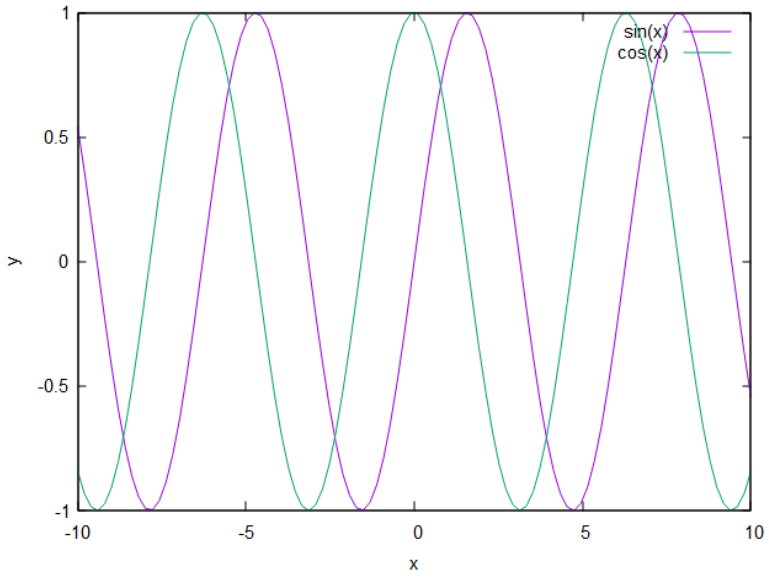
2. **I am not so sure how to do it if they are in different locations.**

I) Plotting functions with gnuplot during C++ program execution (using pipe)

1. There could be various ways to implement gnuplot from C++.
2. One of the most popular ways is to use **pipe**.
3. A common use of a pipe is to send data to or receive data from a program being run as a subprocess.
4. Below is an example of C++ code to plot sin(x) and cos(x):

```
#include <iostream>
#include <stdio.h>    // to use popen & pclose
using namespace std;
int main()
{
    FILE *pipe = popen("gnuplot -persist", "w");

    fprintf(pipe, "set xrange [0:6.284] \n");
    fprintf(pipe, "set yrange [-1.1:1.1] \n");
    fprintf(pipe, "set xlabel 'Deflection' \n");
    fprintf(pipe, "set ylabel 'Col-Force/Beam-Force' \n");
    fprintf(pipe, "plot sin(x) w l, cos(x) w lp  \n");
    fflush(pipe);

    pclose(pipe);      // close the pipe

    return 0;
}
```

which gives output as follows:



5. Note that there are a few things related to the C++ program above which are **(please don't worry too much of you don't understand any of these)**:
   - The library *stdio.h* is needed for *popen*;
   - The *popen* opens a process by creating a pipe streams;
   - The *–persist* keeps the plot open even after the C++ program terminates;
   - The mode argument *w* allows us to write to the stream to send data to the standard input channel of the subprocess;
   - The *fprintf* sends formatted output to a stream.
   - The backslash *\n* for each line is a new line command and it is compulsory to be present to separate each line from the next line.

- The *fflush(pipe)* will flush the pipe, so that after every plot command, a plot is actually generated (or an existing one refreshed) immediately.
- Please not that every

J) Plotting data with gnuplot during C++ program execution (using pipe)

1. Imagine that we have two arrays x and y calculated from a C++ program, and we want to plot these arrays in the form of a graph in gnuplot.
2. For example, let x contains N=101 equally spaced values from 0 to 0.6. Then we can denote the elements of x as x[i]= 0.1i, where i=0,1,2,…,N. Let's denote the elements of y as the negative exponential values of x, that is, y[i] = exp(-x[i]). Assume that calculations of x and y are made during C++ program execution.
3. There are at least two ways to do it, which have been discussed in here and here. These two methods are:
   - Write the data to a temporary file and use gnuplot to plot the data in the file.
   - Directly put the data in the pipe without the need to use a temporary file.

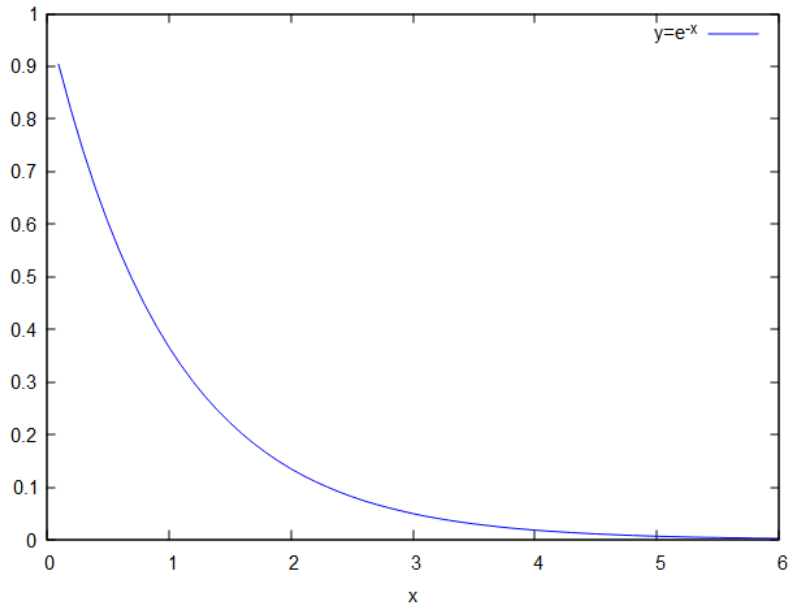
i. Using temporary file

- In this case, we can create an output file using *fstream*, then use pipe to call gnuplot from C++ program. Gnuplot will then plot the data stored in output file.
- The C++ program is as shown below:

```
#include <iostream>
#include <fstream>
#include <cmath>
#include<cstdlib>     // needed for exit()
#include <stdio.h>    // to use popen
using namespace std;
int main()
{
    FILE *pipe = popen("gnuplot -persist", "w");  // initiate the pipe

    ofstream outFile;
    outFile.open("temp.txt");   // open the file
    if (outFile.fail())
    {
        cout << "The file was not successfully opened\n";
        exit(1);
    }

    const int N=61;
    double x[N], y[N];
    x[0] = 0;
    y[0] = 1;

    for(int i=1; i<N; i++)
    {
        x[i] = 0.1*i;
        y[i] = exp(-x[i]);
        outFile <<  x[i] << " " << y[i] << endl; // save data in file
    }

    outFile.close();   // close the  file

    fprintf(pipe, "set xrange [0:6] \n", "set yrange [0:1] \n");
    fprintf(pipe, "set xlabel 'x' \n", "set ylabel 'y' \n");
    fprintf(pipe, "plot 'temp.txt' u 1:2 t 'y=e^{-x}' w l lc 'red' \n");

    fflush(pipe);
```

```
    pclose(pipe);       // close the pipe

    return 0;
}
```

- When we run the above C++ program, we will get a plot as follows:



- Other than using fstream, we can use *fopen* from *stdio.h* to do exactly the same thing as done above.
- The code is as follows:

```
#include <iostream>
#include <cmath>
#include <stdio.h>  // to use popen
using namespace std;
int main()
{
    FILE *pipe = popen("gnuplot -persist", "w"); // initiate the pipe
    FILE *temp = fopen("temp.dat", "w");         // open the file

    const int N=61;
    double x[N], y[N];
    x[0] = 0;
    y[0] = 1;

    for(int i=1; i<N; i++)
    {
        x[i] = 0.1*i;
        y[i] = exp(-x[i]);
        fprintf(temp, "%f %f\n", x[i],y[i]);   // save data in file
    }

    fclose(temp);       // close the file

    fprintf(pipe, "set xrange [0:6] \n", "set yrange [0:1] \n");
    fprintf(pipe, "set xlabel 'x' \n", "set ylabel 'y' \n");
    fprintf(pipe, "plot 'temp.dat' u 1:2 t 'y=e^{-x}' w l lc 'blue' \n");

    fflush(pipe);
    pclose(pipe);       // close the pipe
```

```
    return 0;
}
```

- It is important to note in both cases the file must be closed before the pipe. Through trial-and-error, I found that failure to satisfy this requirement will result in error.

ii. Directly put data in pipe without using temporary file

- I prefer this method, since my C++ program is shorter and less complicated and I don't waste space unnecessarily in my computer to create a temporary file.
- The program is as follows:

```
#include <iostream>
#include <cmath>
#include <stdio.h>  // to use popen
using namespace std;
int main()
{
    FILE *pipe = popen("gnuplot -persist", "w");

    const int N=61;
    double x[N], y[N];
    x[0] = 0;
    y[0] = 1;

    for(int i=1; i<N-1; i++)
    {
        x[i] = 0.1*i;
        y[i] = exp(-x[i]);
    }

    fprintf(pipe, "set xrange [0:6] \n", "set yrange [0:1] \n");
    fprintf(pipe, "set xlabel 'x' \n", "set ylabel 'y' \n");

    fprintf(pipe, "plot '-' t 'y=e^{-x}' w l lc 'blue' \n");
    for (int i = 0; i<N; i++)
        fprintf(pipe,  "%f %f\n", x[i], y[i]); \\ send data to gnuplot
    fprintf(pipe, "e\n");

    fflush(pipe);
    pclose(pipe);      // close the pipe

    return 0;
}
```

- Note that the dash `'-'` after the `plot` is necessary, hence must not be omitted.
- If we want to change certain properties of the gnuplot's figure, we can supply the value to gnuplot through pipe using *fprintf*, which actually has been done above when we send x and y to gnuplot.
- For example, both codes below are equivalent (refer to here where the example is taken from).

```
fprintf(gnuplotPipe, "max='1500'\n");

int maximum = 500; // taken from user input maybe
fprintf(gnuplotPipe, "max=%d\n", maximum);
```

K) Other methods and programs for plotting

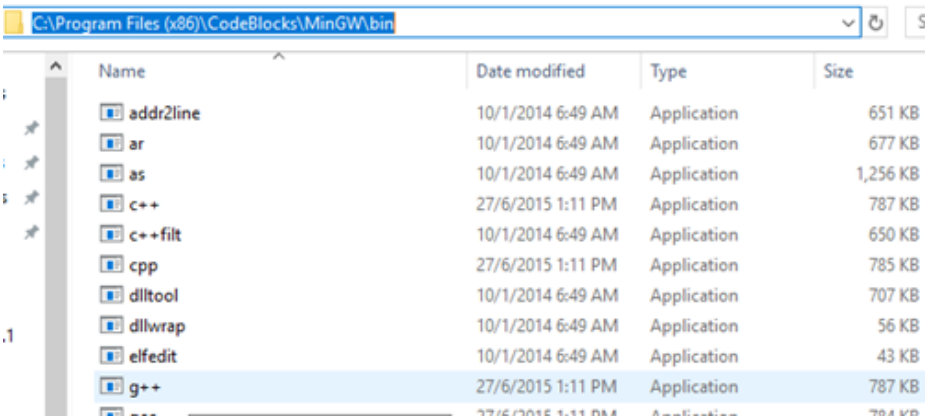
1. There are many other methods to use gnuplot from C++. Amongst them are:
   - *gnuplot-iostream interface* – just an iostream pipe to gnuplot with some extra functions for pushing data arrays and getting mouse clicks. Refer to here for more information. I saw in the example that this method looks easier to use.
   - *gnuplot-cpp* – Refer to here for more information. This method uses pipe.
   - *gnuplot.h* – Refer to the video here for more information. The method is similar to *gnuplot-cpp* and uses pipe.

2. It is also important to mention that we can use many other programs to produce plots from C++. Amongst them are:
   - *PLplot* – Refer to here.
   - *OpenGL* – Refer to here.
   - *Plotutils* – Refer to here.
   - *Matplotlib* – Refer to here and here. This requires *matplotlib-cpp*, a C++ wrapper for Python's matplotlib plitting library, and looks very easy to use.
   - *Qt* – this is made easier by Qwt (2D), QwtPlot3D (3D) and QCustomPlot. As mentioned in here, *"QCustomPlot allows either GPL or commercial. Qwt uses the LGPL license. This is something that you must consider based on the project you're working on"*.

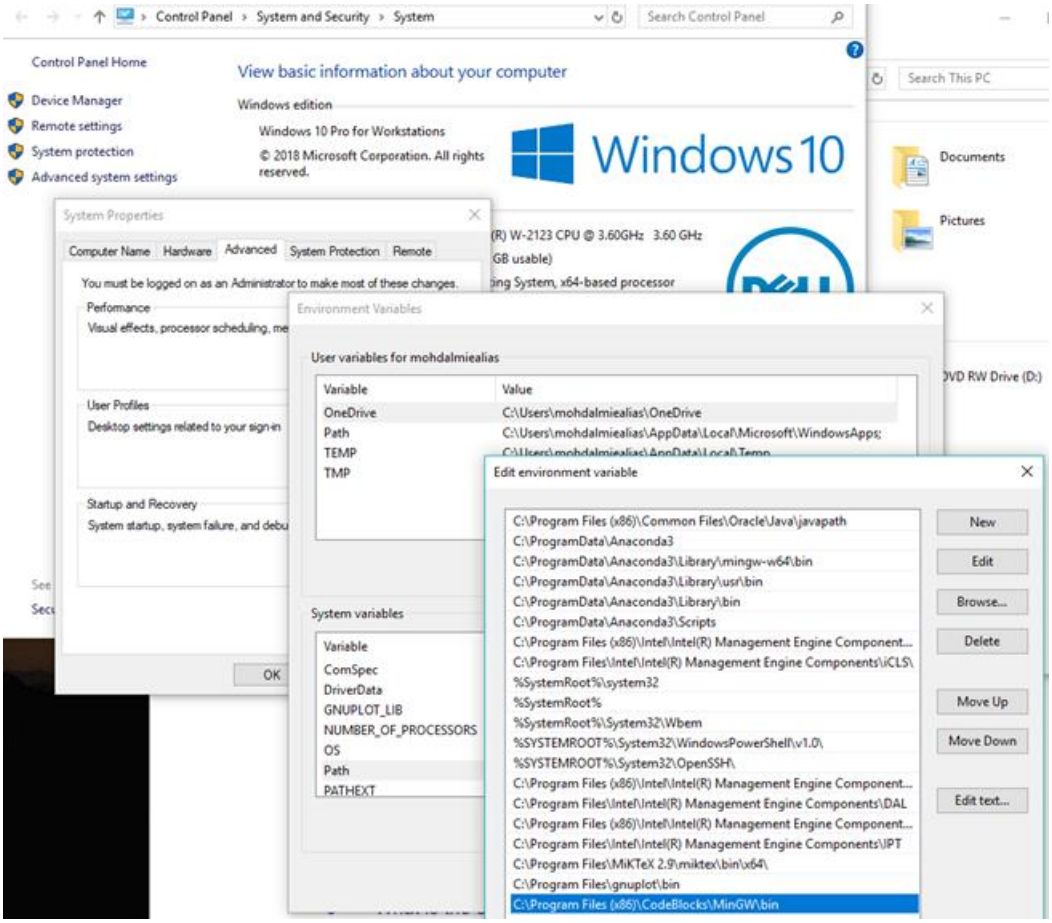
3. Finally, it might be quicker and easier to use interpreted languages such as Python and R. There are a lot of discussion about which programming languages to choose. It is up to you to decide.

L) Extra: Running C++ from command line (in windows)

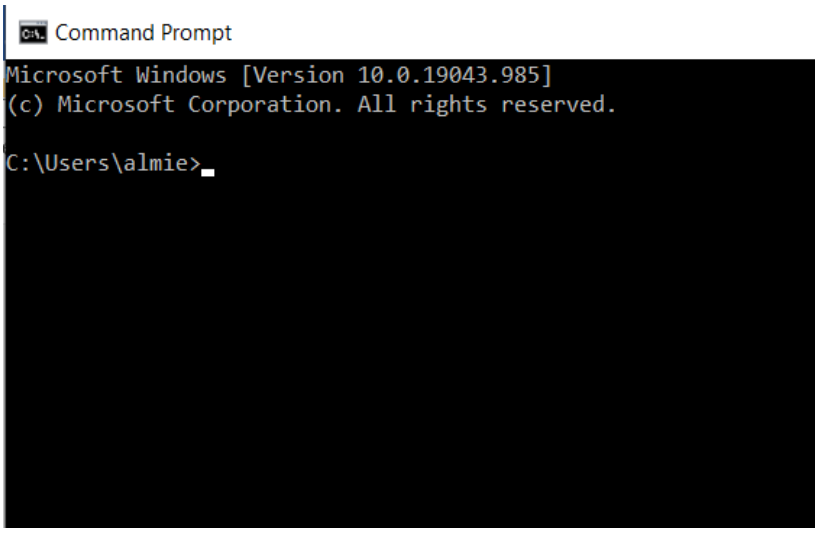
1. We can run C++ from *cmd* or *command line*.
2. First, find the location of C++ compiler (called *g++*).
3. Say that *g++* was downloaded along with Codeblocks, so the location is given in the highlighted text below.



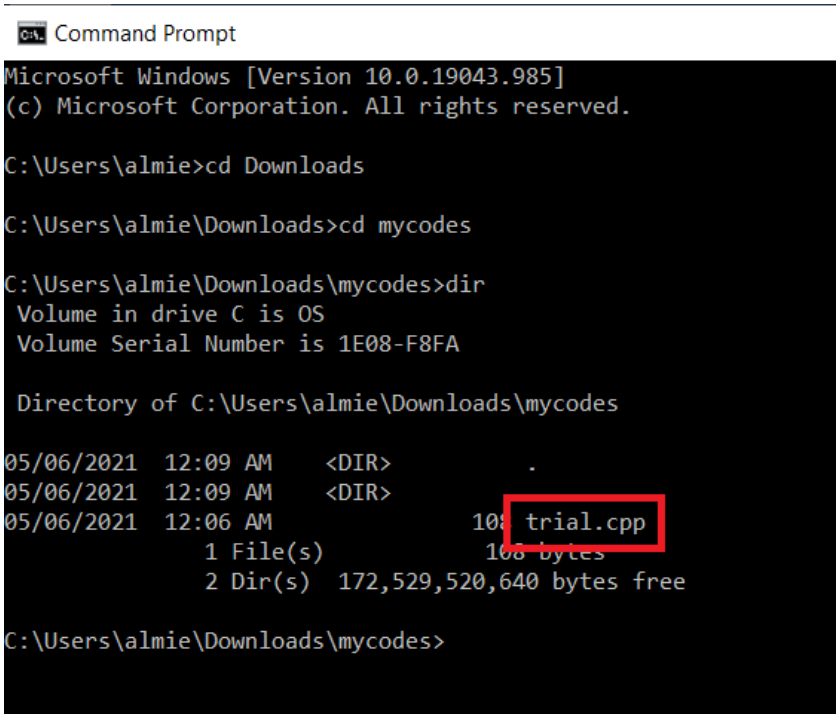
4. Add the location of *g++* to the path.



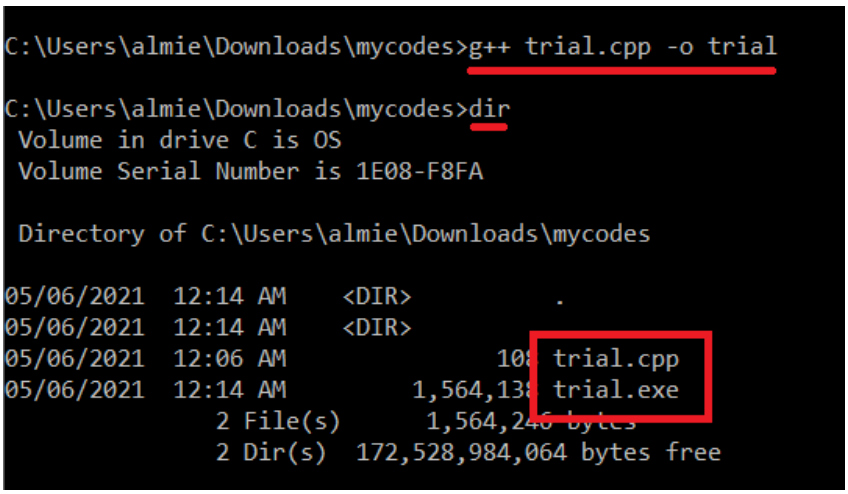
5. Done!
6. Now, we can try check if the process is successful.
7. Open the *cmd* by hitting *Windows* + *S* followed by typing *cmd* (or try search for *cmd*). This will give a display as follows:



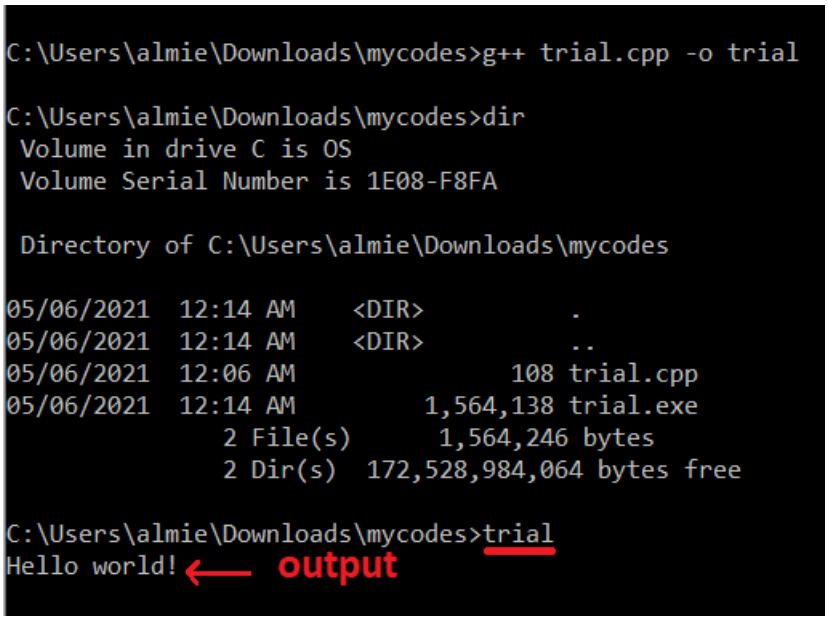
8. Change to the prompt's location to the location of our C++ file. This can be done by using *cd* to change directory. We can use the command *dir* to check the available files in the directory. The process is shown below.



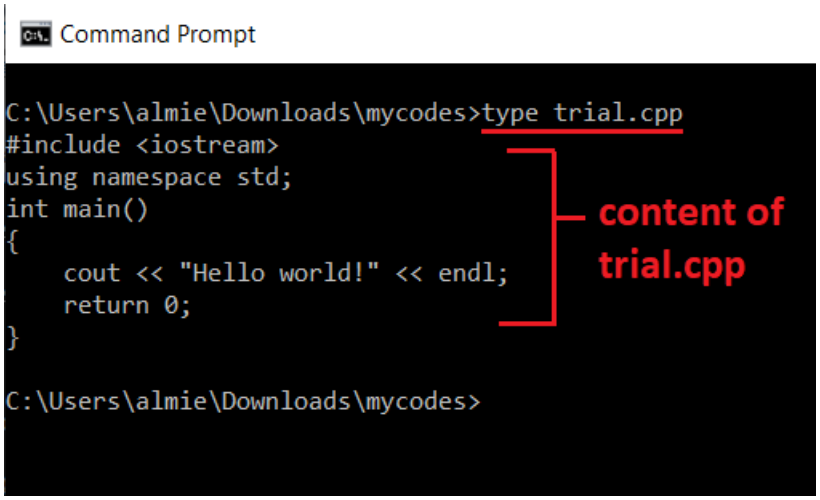
9. Now we can see that there is only one file, named *trial.cpp* in the path *C:/Users/almie/Downloads/mycodes*.
10. To run *trial.cpp* (which is a file that contains command to print Hello world!), we just type *g++ trial.cpp -o trial*. This tells the GNU C+ compiler (*g++*) to take the input file *trial.cpp* and make an output file called *trial.exe* (the *-o* is for output). Again, we can view the files by typing *dir*.



11. Finally, to get the output, type *trial.exe* or just "trial". The output will appear on the cmd.



12. Note that, we can view the content of a file from the *cmd* by using the command *type* (use *cat* in Linux/Unix). This helps us to save time since we don't need to open Notepad, Codeblocks or other editors to view the content. The method to use *type* is shown below.



******* END OF DOCUMENT *******

Updated: 5th June 2021, 12.28am.